Evaluating High Performance Computing Systems at the Naval Research Laboratory

Wendell Anderson, Jeanie Osburn, and Robert Rosenberg
US Naval Research Laboratory (NRL-DC), Washington, DC
{osburn, robert.rosenberg}@nrl.navy.mil and
wanderso@cmf.nrl.navy.mil

Marco Lanzagorta
ITT Corp., Washington, DC
marco.lanzagorta@nrl.navy.mil

Abstract

As a leading edge center within the High Performance Computing Modernization Program (HPCMP), the Center for Computational Science (CCS) of the Naval Research Laboratory evaluates new high performance computing (HPC) assets. The center is currently evaluating two systems - an Altix 3700 running the SUSE Linux Enterprise Server (SLES) 10 operating systems and a Cray XD1. With respect to the Altix system, Naval Research Laboratory is looking at applications requiring large memory and testing the new Intel 10.0 compilers. For the XD1, we are examining the applicability of dual core processors and Field Programmable Gate Arrays (FPGAs) to HPC applications.

Keywords: HPC, XD1, FPGA, Altix

1. Introduction

Since its inception, the CCS of the Naval Research Laboratory (NRL) has functioned at the cutting edge of HPC. Part of its role has been as a leading edge center within the HPCMP of the US Department of Defense (DoD). Currently the CCS is operating super computers from Cray (an XD1) and Silicon Graphics (several Altix 3700's). This paper will describe our experiences with these machines and the work that is being done on them.

2. Cray XD1

The CCS replaced its Multi-threaded Architecture (MTA-2) with an XD1 in September 2005[1] and upgraded it to the current configuration in June 2006. Since that time it has been used for running scientific applications and researching the role of multi-cores and Field Programmable Gate Arrays (FPGAs) in the HPC environment.

2.1. Hardware

The NRL Cray XD1 consists of thirty-six chassis with six nodes in each chassis. Each node of the NRL system consists of two Opteron 275 2.2 GHz dual core processors with 8GBs of shared memory and a 73GB 10K rpm 3.5 in. SATA drive. The full XD1 system thus contains 864 cores with a cumulative raw speed of 3.5 teraflops. The XD1 system has 144 Xilinx Virtex™-II and 6 Virtex™-4 FPGAs.

In addition to the local disks on each node, the XD1 has 30 terabytes (TBs) of fibre channel external disk storage. This space is available from any of the processors via the Lustre® disk system. The Lustre® disk system provides a high bandwidth option for saving large amounts of data.

2.2. Software

Each node of the XD1 runs a Cray modified version of the SUSE Linux kernel. The XD1 has both the gnu and Portland Group Fortran and C/C++ compilers available. For performance reasons most codes are compiled and linked with the Portland Group compiler. Message Passing Interface (MPI) support is provided through mpich 2.6. Users may improve the performance of their applications by using the tuned AMD Core Math Library (ACML) or the Cray Scientific Library. The ARPACK and PARPACK software libraries for finding eigenvalues of large matrices are also installed on the system.

User access to the XD1 is available through a node dedicated to logins. One additional node is devoted to monitoring the XD1 and four more nodes are dedicated to supporting the Lustre® disk system. The remaining 210 nodes are compute nodes that are available to users only via the PBSPro batch queuing system. Currently we are running two queues on the XD1, a small queue that contains the six nodes with attached Virtex™ 4's and a second large queue that contains the other compute nodes.

0-7695-3088-5/07 $25.00 © 2007 IEEE

Programming support for the FPGA is provided through the standard Xilinx software and through three higher order languages: Mitrion's Mitrion-C, Celoxica's Handel-C, and DSPlogic's Simulink based package. The first two packages run natively on the XD1, while the last package only runs on a Windows PC (the code generated for the FPGAs must be transferred from the PC to the XD1).

2.3. HPC Scientific Applications

The XD1 has proven to be a popular resource among the researchers that use the NRL facility with over 3.3 million node hours of computational time used since the initial installation of the system. The top six codes in terms of node hours are given in Table 1.

Table 1. Code Usage by Time

Code	Node Hrs
ARMS	1,350,000
NOZZLE	800,000
NRLMOL	600,000
ADF	120,000
CHARMM	90,000
STARS3D	80,000

By far the largest usage is by the Adaptive Refined Magneto-hydrodynamic Solver (ARMS)[2], a massively parallel, flux-corrected transport based code built upon Message Passing Interface communications and NASA Goddard's PARAMESH parallel adaptive meshing toolkit. The ARMS code performs three-dimensional, time-dependent simulations of solar magnetic storms. Dr. C.R. DeVore and Dr. S.K. Antiochos are using the code to simulate solar storms resulting from a solar eruption. Dr. Judy Karpen is using ARMS to test the theory that the concentration of explosive solar activity in filament channels is a result of flux cancellation driven by convective motions beneath the solar surface concentrating the shear in such sites.

The second largest usage is NOZZLE, an MPI code that solves the compressible Navier-Stokes equation on structured multi-block domains using a domain decomposition model for parallel processing. Dr. Andreas Gross is using the NOZZLE program to support Air Force Programs that require the numerical simulation (with or without turbulence) of Coanda wall jet experiments.

The NRLMOL code[3] is an NRL developed code that implements the Density-Functional formalism for clusters and molecules by using MPI to parallelize the problem by using a master–slave architecture. Dr. Tunna Baruah and Dr. Mark Pederson have been using this code on the XD1 to study molecular vibrational effects on the simulation of a light-harvesting molecule[4].

The Amsterdam Density Functional (ADF) package[5] (http://www.scm.com/) is a commercial universal density code for chemists. Dr. Stefan Badescu and Dr. Victor Bermudez of NRL are using the code to perform a quantum-chemical analysis of the interaction of chemical warfare agents with materials.

The Chemistry at HARvard Macromolecular Mechanics (CHARMM) code (http://www.charmm.org/) is a program for macromolecular simulations, including energy minimization, molecular dynamics and Monte Carlo simulations. Dr. Alex MacKerell and Deva Priyakumar use CHARMM to study the interaction of urea with P5GA RNA.

The STARS3D[6] code is a frequency-domain parallel code for three-dimensional structural/acoustic/seismic simulations. It is based on a high-order finite element method and incorporates several advanced numerical features like hierarchic basis functions, infinite elements, perfectly matched layer approximations and domain-decomposition (FETI) solver. Dr. Seikat Dey is using the code to study wideband acoustic radiation and scattering from submerged elastic structures.

2.4. FPGA Accelerated Applications

Ken Rice and Tarek M. Taha of Clemson University are working on accelerating large-scale models of the neocortex, implementing George and Hawkins'[7] recognition algorithm in hardware. This model utilizes a network of nodes—where each node implements Pearl's[8] Bayesian belief propagation. At present the application has modeled up to 321 nodes using a total of 64 of the XD1's Xilinx-II FPGAs.

NRL is currently working with Mitrion to port their SGI RASC FPGA BLASTN implementation to the Cray XD1. This work consisted in changing the Mitrion code that uses 128 bit data paths from a paired set of QDRAM banks on the SGI to 64-bit data paths from a single QDRAM. It also required modifications to the Cray FPGA –XD1 interface. Dr, Anthony Malanoski of NRL is planning on using this BLASTN implementation to determine organism identification from relatively short sequence.

Commander Charles Cameron of the United States Naval Academy has been using the XD1 to evaluate ray tracing software for lens design and general optical systems processing He has used an MPI C-program to study a system with 22 planar surfaces, two paraboloid reflectors, and one hyperboloid refractor. In going from one core to 839 cores, he achieved an efficiency of 97.9%. His next step is to perform the ray tracing on the XD1 FPGAs.

2.5. Dual Core Technology

As part of our evaluation of the XD1, we have run MPI codes of interest to the Department of Defense (DoD) using only one core per processor and using both cores of a node[1]. Results of eleven applications are given in Table 2.

Table 2. Dual Core Efficiency

Application	One Core	Both Cores	Efficiency %
STATIC	313	450	56
CAUSAL	275	293	93
LANCZOS	771	1371	22
NRLMOL	14283	16260	90
ARMS	2090	2524	79
NOZZLE	27498	27286	101
CHARMM	233	254	91
AVUS	1197	963	120
HYCOM	823	849	97
OOCORE	5274	7716	54
RFCTH2	279	448	39

Since the one core and two core cases were run over the same number of processors, the running times assuming no memory contention should have been the same. In order to determine how close we came to obtaining this ideal, the efficiency is given by

$$100*(1-(T_4-T_2)/T_2)$$

where T_2 is the wall clock time for running the code on N nodes using one core per processor and T_4 is the wall time running it on N/2 nodes using all the cores on the nodes. Since the same number of processors are used in both cases, if the times T_2 and T_4 are the same, we have perfect efficient (100%). If T_4 is twice T_2 then there is no advantage as running two copies of the application, each on N/2 nodes would finish at the same time as running the application consecutively on N nodes using only one-half of the cores and the efficiency is 0%. The worst case was LANCZOS a sparse matrix solver that had a high ratio of memory accesses to floating-point operations and a low cache reuse. Over half of the applications had efficiencies of 90% or better with two exceeding 100%, probably a result of the processors being closer together in the dual core case.

2.6. Lustre® Disk Input/Output (I/O)

The running time of applications on high performance computers can be significantly influenced by the time it takes to move data between memory and disk. Users may need to read input data, write out calculated results, store and retrieve data on temporary scratch space, and save restart files

Each line of Table 3 below lists the read and write rates for I/O from one core on each node. For the NRL configuration, the peak I/O rates were nearly reached on eight nodes for reads and 4 nodes for writes. Using more nodes for I/O did not result in any significant increase in I/O bandwidth.

Table 3. Lustre® I/O rates

Nodes	Read *(MB/sec)*	Write *(MB/sec)*
1	156	417
2	326	821
4	630	1298
8	1224	1393
16	1460	1250
32	1420	1280

3. SGI Altix Systems

For the past several years NRL has been evaluating the SGI Altix systems. In December 2003, NRL received the first Altix 3700 within the HPCMP and performed the evaluation of the system[8] before they were obtained by the Major Shared Resource Centers. Additional systems have been added to the center over the years. Currently we are engaged in a variety of evaluations including the Intel beta compiler, large memory applications, and the CSFX and Lustre® shared file systems.

3.1. Hardware

Currently the Center for Computational Center has four large systems on the floor. A summary is given in Table 4.

Table 4. NRL Altix HPC Machines

Machine	Name	Chip Speed *(GHz)*	Cores	Memory *(TB)*
Altix3700	niobe	1.3	128	0.25
Altix3700	morpheus	1.6	256	2.0
Altix3700	neo	1.6	128	0.5
Altix4700	seraph	3.0	256	1.0

Niobe is available to users as an unallocated production machine for users in the HPCMP. The other three machines are research machines—morpheus for evaluation of codes that require a large amount of memory, neo for visualization of data, and scraph for research in dual core and FPGA technologies.

3.2. Software

The machines are all running the SGI SUSE-based SLES operating system. All of the machines except niobe (SLES 9) are running SLES 10. The Intel 8 and 9 FORTRAN and C++ compilers are available for developing code. Both niobe and morpheus run the PBSPro batch queuing system and user jobs are run in batch mode on these machines. Jobs on the other machines are run interactively.

Home directories are in an AFS cell that is available from any of the machines. Niobe and Morpheus have local scratch disk systems. CXFS™ scratch space is also shared among all of the machines. Archival storage is provided by an SGI SAN system.

3.3. Intel 10.0 Beta Compiler

The beta testing of the Intel 10.0 FORTRAN compiler was performed in two phases During the first phase, regression testing was performed to ensure that our codes still ran and had no significant performance degradations. The second phase was to investigate how the automatic parallelization worked on shared memory codes.

The first step in our evaluation was to recompile with the switches used in the acceptance of the Altix system, run and time the parallel TI-05 benchmarks. The codes in this suite include AVUS, GAMESS, HYCOM, OOCORE, OVERFLOW2, and RFCTH2. Using the same switches as supplied with the TI-05 distribution, with the exception of RFCTH2 all of the codes compiled, linked, and ran correctly. Running times for the programs were within a few percent of the times for the codes using the old version 9 compilers.

One of the files (p3cut1.F) of RFCTH2 aborted with a Severe ** Internal compiler error ** when compiled with the original beta compiler. Subsequent releases of the compiler compiled the code correctly. Running the code with the latest release resulted in correct answers with no significant change in the running time.

The next step was to test the compiler on several production codes (ARMS, NRLMOL, FAST3D, CHARMM, and CAUSAL) that are run at NRL. All codes compiled, linked, and ran correctly. With the exception of ARMS all codes ran within a few percent of the running times when compiled with the version 9 compiler. In the case of ARMS, a 25% decrease in running time of the program was noted.

After completing the regression testing, we tested the parallelizing features of the codes. The shared memory code FLUX was chosen for the evaluation.

FLUX calculates the temperature dependent electronic excitation spectra and super conducting densities of materials by iteratively solving over a four dimensional grid the non-linear equations of propagator functional theory, Dyson's equation and a self-iteration equation. Fast Fourier Transforms (FFTs) are used to transform between the position-imaginary time and crystal momentum-imaginary frequency spaces. A frequency conditioning method removes unphysical artifacts of the FFT procedure and ensures that the solution is self-consistent. Most of the time is spent in updating the equations at each grid points and performing multiple FFTs over each of the axis of the grid. The value at each grid point is a function of the grid point for the previous iteration and none of the current grid points. Both the updates and FFT calculations are embarrassingly parallel. The Flux code is quite large containing 9,500 lines of code and almost 400 DO loops. Trying to insert OpenMP directives is a formidable task so being able to automatically parallelize the code would be very valuable

Runs were made on 32 processors with auto parallelization turned off (serial) and on (parallel) and with the FFTs performed using the Cray Scientific Libraries cmsl (serial) and cmsl_mp (parallel). The results of running the code on 32 processors under the four scenarios are given in Table 5.

Table 5. FLUX Intel 10.0 Running Times

Grid Updates	FFT	Time *(secs)*
Serial	Serial	878
Serial	Parallel	799
Parallel	Serial	411
Parallel	Parallel	85

These results clearly demonstrate the speedup achievable by the Intel 10.0 compiler automatic parallelization.

3.4. Large Memory Scientific Applications

The two terabyte 256 processor Altix 3700 provides an ideal platform for testing and evaluating the performance of scientific codes that require a large amount of memory. The CAUSAL code developed by Guy Norton[9] and extended from two-dimensional (2D) to three-dimensional (3D) is one such code.

The CAUSAL code accurately models pulse propagation and scattering in a dispersive medium. CAUSAL first models the medium as non-dispersive by use of a fourth order in time and space 3D Finite Difference Time Domain scheme representation of the "modified equation approach" of the linear wave equation[10]. At each dispersive grid point, the wave equation has been modified by an additional term (the derivative of the convolution between the causal time domain propagation factor and the acoustic pressure). Absorbing Boundary Conditions (ABCs) were implemented via the Complementary Operators Method a

differential equation-based ABCs that reduces the reflection of the pulse when encountering a grid boundary.

The test case required almost 150 million grid points (1.2 TBs) over a four-dimensional grid in space and time. The first part of the code set the value at each grid point to zero. The initial runs under SLES 9 of the SGI modified SUSE Linux kernel required over an hour to initialize the arrays. This slowness is due to known deficiencies in the scalability of the routine to allocate memory on the machine. The same test case was rerun after the machine was upgraded to SLES 10. The initialization time was reduced to ten minutes—still much slower than desired but acceptable. Currently SGI has no announced plans to provide further improvements in the memory allocation.

3.5. TimesTen

Oracle's TimesTen is an application-tier database and transaction management built on a memory-optimized architecture accessed through industry-standard interfaces. In the Fall of 2006, NRL's two-terabyte memory ALTIX provided a large memory system for demonstrating the capabilities of this application on a large SQL RDBMS database. The results demonstrated four orders of magnitude improvement over a conventional disk-based RDBMS system with over 1 billion records ingested per hour. According to Winter Corporation's 2005 Top Ten VLDB Survey Furthermore, this study, with its 10 billion rows in a single Linux database, ranked as the second largest database (including disk-based) in the world.

3.6. Lustre®

Recently we have started to investigate the use of the Cluster File Systems, Inc (CFS) Lustre® file system. Although upcoming versions of Lustre® (1.8/2.0) have support for using Kerberos for authentication between clients and servers, limitations were noted. While all of the necessary protocol work has been completed, the client-side implementation has a few deficiencies that make it impractical to be used by sites that rely heavily on Kerberos. NRL will be working with CFS to make the client-side portion of the Lustre®' Kerberos authentication more practical for sites that use Kerberos. Also we will be investigating the performance impact of Kerberos authentication in a Lustre® environment.

4. Conclusions

The XD1 has proven to be a popular resource for our scientific community. Our researchers are now beginning to take investigating the use of FPGAs in the HPC world. The learning curve for the FPGA higher order languages is significant. While Cray has discontinued the XD1 as a product line, they continue to support the product, although bug fixes are limited to those that do not require a major rework of the software. We expect that the XD1 will be available as part of our Center for at least the next three years.

The Altix systems provide us with platforms to test and evaluate a variety of leading edge problems including large memory applications and shared file systems.

Acknowledgements

We wish to thank all of the scientists mentioned through out the report for the use of their codes and willingness to answer all of our questions. Ray Yee of Cray researched many of our XD1 questions and Jace Mogill of Mitrionics who ported the Mitrion BLASTN FPGA implementation from the SGI RASC implementation to the XD1. J. Barton of SGI provided insights into the allocation of memory by the SLES 9 and 10 operating systems. Pascal Girard and Gerry Kolosvary of Fedccntric and John Conway of Oracle were responsible for getting TimesTen installed and running on our SGI machines. Ken Hornstein of ITT Corp is developing Kerberos modifications of the Lustre® file system.

References

1. Anderson, W., M. Lanzagorta, R. Rosenberg, and J, Osburn, "Early Experiences with the XD1 at the Naval Research Laboratory." *Cray Users Group Meeting*, Lugano, SW, May 2006.

2. Devore, C. Richard, Peter J. MacNeice, and Kevin M. Olson, "An Adaptively Refined MHD Solver for Solar, Space, and Astrophysical Simulations." *DCOMP Meeting 2001* Scientific Program.

3. Pederson, M., D. Porezag. J. Kortus, and D. Patten, "Strategies for massively parallel local-orbital-based electronic calculations." *Physica status Solidi*, KnB217 197, 2000.

4. Baruah, T., M. Pederson, and W. Anderson, "Massively Parallel Simulation of Light Harvesting in an Organic Molecular Triad." *DoD HPCMP Users Group Meeting*, Nashville, TN, June 2005.

5. Velde, G. te and F.M. Bickelhaupt, et al, "Chemistry with ADF." *Journal of Computational Chemistry*, Vol. 22, No. 9, pp. 931–967, 197, 2001.

6. Dey, S. and Dibyendu K. Datta, "A parallel hp-FEM infrastructure for three-dimensional structural acoustics." *International Journal for Numerical Methods in Engineering*, Vol. 68, Issue 5, pp. 583–603, 2000.

7. George, D. and J. Hawkins, "A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex." In *International Joint Conference on Neural Networks*, 2005.

8. Pearl, J., *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufman Publishers, San Francisco, CA, 1988.
9. Norton, G.V. and J.C. Novarini, "Including dispersion and attenuation directly in the time domain for wave propagation in isotropic media." *J. Acoust. Soc. Am.*, 113, pp. 3024–3031, 2003.
10. Cohen, G., *Higher-Order Numerical Methods for Transient Wave Equations*, Springer, pp. 35–63, 2001.